Self-Repairable GALs

Chong H. Lee, Douglas V. Hall, Marek A. Perkowski, and David S. Jun
Portland State University, Dept. of Electrical & Computer Engineering

Abstract

This paper describes the concept of self-testable and self-repairable GAL (Generic Array Logic) devices for high security and safety applications. A design methodology is proposed for self-repairing of a GAL which is a kind of EPLDs (Electrically Programmable Logic Devices). The fault-locating and fault-repairing architecture with electrically re-configurable GALs is presented. It uses universal test sets, fault-detecting logic, and self-repairing circuits with spare devices. The design method allows to detect, diagnose, and repair of all multiple stuck-at faults which might occur on E^2CMOS cells in programmable AND plane. A self-repairing methodology is presented, based on our design architecture. A "column replacement" method with extra columns will be introduced that discards each faulty column entirely and replaces it with an extra column. The evaluation methodology proves that the self-repairable GAL will last longer in the field. It gives also information how many extra columns a GAL needs to reach a lifetime goal in terms of simulation looping time until a GAL is not useful any more. Therefore, an ideal point could be estimated, where the maximum reliability can be reached with the minimum cost.

1. Introduction

PLDs (Programmable Logic Devices) have become extremely popular in modern systems since they can be reprogrammed simply and inexpensively without making expensive and time-consuming PCB (Printed Circuit Board) changes as was required by the earlier designs that used random logic ICs [1,2,3]. They use various programming technologies like fusible link, E^2CMOS (Electrically Erasable Complementary Metal Oxide Semiconductor), and others. PLDs allow implementation a variety of logic circuits using EDA (Electronic Design Automation) tools. However, as the complexity of digital devices increases and the chips' geometry shrinks, the probability of having faulty components (input/output lines, and product terms) also increases [4].

Thus, in-circuit testing of PLDs, especially PLAs (Programmable Logic Array) has become of primary importance and has attracted the attention of large research community [5,6,7,8,9,10]. In high reliability applications, the circuit should test itself in real time and repair itself as early as possible. This requires special logic built into a chip to make it easily testable and diagnosable, as well as self-repairable by means of hardware re-programmability. When the high reliability and quick in-field repair of a digital system is of the primary importance, the self-testing system cannot be sufficient and the self-repairing system would be desired. As far as we know, nothing has been previously published on self-repair of GALs or similar devices.

Computers and other digital systems are subject to any number of faults caused by inadequate quality control during manufacture, the wear and tear of normal operation, and other. Failures occur in CMOS due to manufacturing defects and due to wear out mechanisms whose effects are accumulated over time [11]. External disturbances

such as heat, radiation, and electrical and mechanical stress produce failures to even higher extent [11].

The failure rate and yield calculation of digital ICs has been surveyed in order to understand better the real problem; actually how and which faults are likely to occur in the real world. Particularly, failure rate of EEPROM is concerned, since our target model, GAL, uses EEPROM technology with E^2CMOS cells in a programmable AND array, as well as an EEPROM structure. Memories are particularly sensitive to aging as a function of cycling.

Now, we refer to the Early Failure Rate (PPM; Parts Per Million devices or DPM; Defects Per Million devices from 0 year to 1 year) and the Long Term Failure Rate (FIT; Failures In Time from 0 year to 10 years) from data provided by National Semiconductor Corporation. Early failure rates were calculated at 60% confidence using the Chi-Square distribution. Long term failure rates were calculated at 60% confidence using the Arrhenious equation at 0.7eV activation energy and derating the stress temperature to an application temperature of 55°C [12,13]. The data used to calculate the failure rates were obtained from high temperature operating life tests (OPL) performed on product qualification and long term audit (LTA) programs during the period from May 4, 1998 to May 4, 1999 [12]:

- 0.35μm CMOS: Early Failure Rate = 194 PPM, FITs = 4.66 %
- 0.50μm CMOS: Early Failure Rate = 141 PPM, FITs = 14.34 %
- EEPROM: Early Failure Rate = 489 PPM, FITs = 5.07 %

The values of the PPM and FIT of EEPROM shown above suggest that it is reasonable to invest in a repair mechanism. These data will be used to simulate our repair algorithm in order to get practical useful information in the evaluation section.

In the present paper, the concept of the self-repairing GAL will be introduced. A self-repairing method, Column Replacement with extra columns will be presented, in which the faults are located with accuracy to each E^2CMOS cell of a programmable AND array in a GAL. The respective faulty elements (E^2CMOS cells/Cross-points) are replaced with the new ones (in extra columns) by automatic reprogramming of the chip. In addition to GALs, this self-repairing method uses a FLFRP (Fault-Locating/Fault-Repairing Processor), diagnosis/repair bus, and the memory that stores fault location and repair-related data. We also propose an evaluation methodology in this paper, which proves that using the column replacement method with extra columns makes lifetime of a GAL longer than using no self-repairing method. This methodology is based on simulating the self-repair algorithm. It shows that lifetime for a GAL that uses our self-repairing method is longer than the lifetime of a GAL that does not use a self-repair method. It gives also information how many extra columns a GAL needs to guarantee a given lifetime.

Failure mode analysis of electronic devices is of utmost importance in the case of high performance systems, such as air and space systems. It is therefore important to establish the expected lifetime and to understand the failure modes of these components for procurement and maintenance purposes. With the trend and demand to develop high temperature tolerant electronics, there is an increasing need for repair methodology not only to assure performance reliability, but also to qualify commercial off-the-shelf components for more critical use that can result in a significant acquisition and maintenance cost saving. If the repair mechanism is used for testing either on the manufacturing process or on outgoing products, the test cost will be decreased and the reliability of the outgoing products will be increased. In

other worlds, it will minimize defect levels at the production test and thereby promote both the cost efficiency and the confidence in the reliability of the outgoing products.

The next section introduces a GAL. The design methodology, test set generation, assumptions of our approach, and fault models are described in section 3. In section 4, a self-repairing method (column replacement with extra columns) is shown with some examples. The evaluation methodology and the results of the evaluation are presented in section 5, and conclusions and future works are discussed in section 6.

2. Introduction to GAL (Generic Array Logic) devices

Lattice Semiconductor company introduced GAL devices such as the GAL16V8 in the mid 1980's. It has a fixed OR array and a programmable AND array. The re-programmable array is essentially a grid of conductors forming rows and columns with an electrically erasable CMOS (E^2CMOS) cell at each cross-point, rather than a fuse as in a PAL [14,15,16]. The programmed state of a simple logic function is schematically shown in Figure 1.

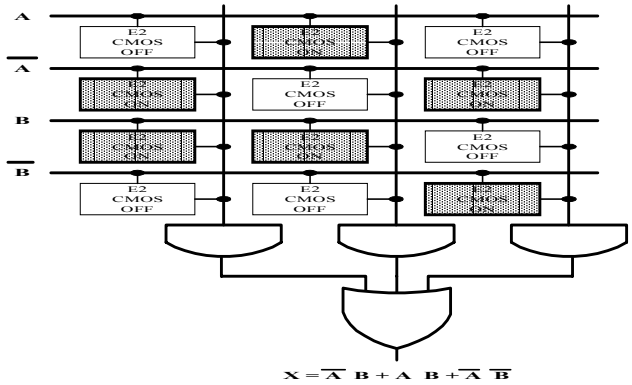


Figure 1. A programmed simple logic function in a GAL

Each column is connected to the input of an AND gate, and each row is connected to an input variable or its complement. Any combination of input variables or complements can be applied to an AND gate to form any desired product term by programming each E^2CMOS cell to be either 'ON' or 'OFF'. A cell that is ON effectively connects its corresponding row and column, and a cell that is OFF disconnects the row and column. The cells can be electrically erased and reprogrammed. The GAL has the programmable logic and the OLMC (Output Logic Macro Cell) logic that includes OR gates and flip-flops [16].

Note that a simple array will be used to graphically describe complex PLD structures: we will use special notation shown below, because a typical PLD has many inputs, outputs, and product terms. The modified AND input lines, likewise OR input lines, are also shown to simplify many input lines of an AND gate as in Figure 2.

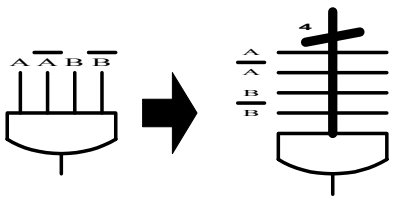


Figure 2. A simplified notation for input lines of an AND gate

The GAL16V8 provides 3.5ns maximum propagation delay, 250MHz clocking, full programmability, low power consumption, and 100 erase/write cycles [16]. The E^2CMOS cell makes our self-repairing methodology possible. Thus, we choose as our model in this paper the GAL16V8, which is a simple low density PLD.

For the more detailed description of pins and configurations the reader is referred to a GAL data book from Lattice Semiconductor, Inc. The sixteen primary inputs which include feedback paths from the OLMC are considered for this project, thus there are 32 input lines, which come with complements of each input variable, in the AND array. The eight primary outputs and eight product terms per an OLMC are considered in our model, thus there are 64 (8X8) product terms in this model. Therefore, the total number of E^2CMOS cells (cross-points) is 2048.

3. Design methodology

In this section, we develop a design methodology for self-repairing an E^2CMOS cell on a programmable AND plane of a GAL, and describe our fault model, assumptions, and an universal test set for detecting and locating faults on each cross-point (E^2CMOS cell).

3.1 Fault model and assumptions

Fault modeling is concerned with the systematic and precise representation of physical faults in a form suitable for simulation and test generation [17]. Such a representation usually involves the definition of abstract or logical faults that produce approximately the same erroneous behavior as the actual physical faults [17]. Good fault models should be straightforward, accurate, and easy to use. The most widely used fault model is the stuck-at fault model, which has been used for fault analysis and test generation in all types of logic circuits [17,18]. While exact coverage figures are difficult to obtain, substantial empirical evidence shows that for general combinational or sequential logic circuits implemented with common MOS or bipolar technologies, the stuck-at fault model provides good coverage of permanent physical faults [17,18]. The most dominant failure modes in CMOS are shorts and opens [11]. In a switch-level representation of a CMOS circuit, MOS transistors are modeled as switches that conditionally transfer signals. The stuck-open fault model assumes that a faulty transistor never switches on (permanently disconnected), while a stuck-on fault model assumes that a faulty transistor never switches off (permanently connected) [11]. The fault model for the memory cell array is presented in [19] and [20]; one or more cells are stuck-at 0 or 1. The functional defect (functional level fault model), which is very useful for describing a wide variety of faults, is introduced in [20]. Functional defects are those that will cause a functional failure or degradation in functional performance either immediately or in the short term [21].

The cross-point stuck-at faults, which are located in E^2CMOS cells, are considered as our fault model because E^2CMOS cells of an AND array form a large percentage of a GAL and E^2CMOS cells' array is the same structure as the EEPROM. Each E^2CMOS cell (cross-point) of a programmable AND array of a GAL, which is located between a vertical line (row, input line) and a horizontal line (column, product term), may be ON or OFF permanently, caused by an aging problem or by other facts referred in section 1. It is called the cross-point stuck-at-1 (simply, s-a-1) if the E^2CMOS cell of a particular cross-point, which should be programmed as OFF, is ON. If the E^2CMOS cell of a particular cross-point, which should be programmed as

ON, is OFF, it is called the cross-point stuck-at-0 (simply, s-a-0). Only these faults will be considered in this paper because we assume that there were no faults found in a GAL after the manufacturing process. Note that horizontal lines and vertical lines represent product terms and data inputs, respectively in data book, but rows will be represented as data inputs and columns will be used as product terms in the rest of figures in this paper.

The following assumptions will be used for the design, self-repair and evaluation methodologies. There are no faults in an initial state after manufacturing GALs. The primary input/output fault does not exist in a GAL, and also every AND gate input line does not have faults. For GALs, the cross-point stuck-at faults are considered much more probable than the stuck-at faults in wires. The cross-point faults (s-a-0, s-a-1) as defined above are only taken into account in this project. If the E^2CMOS cell is ON, binary data '1' will be stored in memory, and if the E^2CMOS cell is OFF, binary data '0' will be stored in memory. In order to replace a faulty column, several extra columns are pre-allocated in each OR gate (OLMC) in a GAL. There are at least two E^2CMOS cells programmed as ON for a pair of input variables, like A and complement of A. When a fault occurs in a certain column of a particular OLMC, this faulty column can be replaced with an extra column only in that OLMC. When an E^2CMOS cell is OFF, it can be called 'programmed as OFF.' This cell disconnects a primary input from an AND gate input, and '1' will be on the AND gate input. It will be described with just an intersection between a row and a column in a logic diagram. When an E^2CMOS cell is ON, it can be also called 'programmed as ON'. This cell connects a primary input from an AND gate input. It is denoted by 'X' between rows and columns in a logic diagram.

3.2 Design architecture

The complete digital system in our approach is a network of blocks realized as separate integrated circuits. Each block is a two-level realization of a Boolean function, and is realized with a GAL. FSMs (Finite State Machines) are composed of Boolean Logic and registers located in OLMCs. This seems to be reasonable, since as the EPLD/FPGA (Field Programmable Gate Array) technology advances, functions and machines of even greater sizes can be implemented in them. Reprogramming of a GAL serves to replace gates in that GAL. The proposed method assumes that the global fault of the block is signaled with go/no-go signal when the redundant area to create gate (i.e. the extra columns) in a block is already exhausted after some repairs, which means there are no more spare columns to be replaced with. The block can be a single module (a chip, a board) or different such modules. In the first case, the module should be replaced, and in the second case, the modules and their connections should be tested independently. We assume that each block includes just one self-repairable GAL. Thus, each block has a programmable AND array with several extra columns and fixed OR (OLMC) plane since it is realized as a GAL.

In this point, we consider the problem of getting the maximum usefulness from each block (GAL) in a system at the lowest level, when the system degraded over time with the new faults arising in the block that have been already tested in the manufacturing process.

The following figure shows the general scheme of the design architecture for a Self-Repairable GAL. It uses extra columns in an AND plane, scan registers (SCR1 and SCR2), diagnosis/repair bus, and a FLFRP which has a MAP (Memory AND Plane), a SAP (State AND Plane), a NC (Next Column) register, three scan registers

(SCR3, SCR4, and SCR5), a Comparator, a Central Fail-Safe maintenance Controller, and Data Path/Addressing Unit. The FLFRP can be realized with a micro-controller.

The SCR1 will have a test vector fed from the controller of FLFRP in a test operation mode. In the normal operation mode, the primary input data are fed into an AND plane instead of the data from SCR1. The SCR2 has a scanning result corresponding to a test vector from the SCR1. This is put on the Diagnosis/Repair bus to be on SAP of the FLFRP. In the normal mode, the product terms of an AND plane are just transparent through the SCR2 into the OR plane. The diagnosis/repair bus is assumed to include control, data, and address bus. The '*i*' denotes the number of primary inputs in an AND plane of GAL. The '*n*' denotes the total number of inputs in an AND plane of GAL, including feedback and complement of the primary inputs. The '*k*' denotes the number of primary outputs in an OR plane of a GAL. The '*m*' denotes the number of product terms (columns) in a GAL. The '*y*' denotes the number of product terms that each OR gate has in an OR (OLMC) plane, thus each OR gate has the same value of '*y*'. The detailed inner structure of the block is shown in Figure 4.

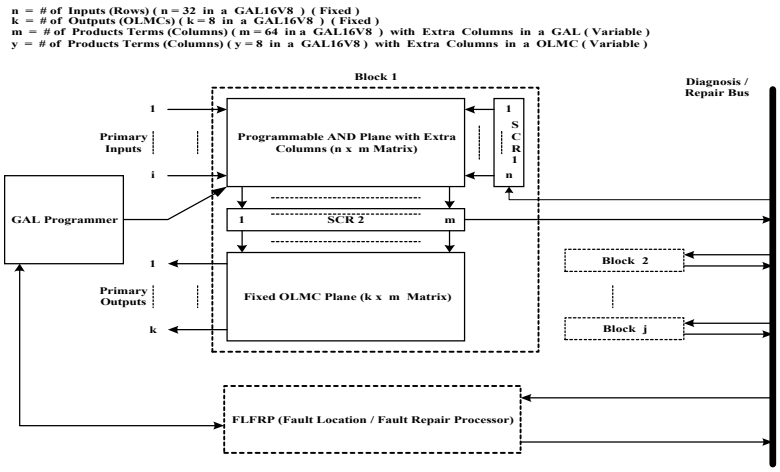


Figure 3. Design architecture for self-repairable GAL

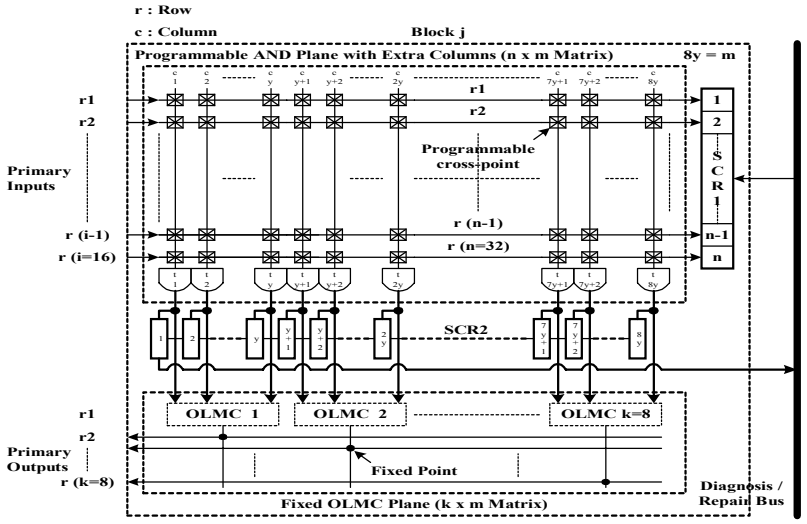


Figure 4. Inner structure of the block

There are '*n x m*' cross-points, which are programmable with E^2CMOS cells in the AND plane. The '*n*' is said to be the number of rows (inputs), and the 'm' is said to be the number of columns (product terms). The '*i* (=16)' includes the feedback input from the OLMCs, thus the total number of rows (inputs) is 32 (= *n*). Each bit of the *n*-bit SCR1 corresponds to 1 row. The extra column means that there are several extra product terms in each OR gate of an OR plane of a GAL. We assume that there is the same number of extra columns in each OR gate, also called the OR group. If there are 8 OR gates in an OLMC plane of a GAL and '*y*' number of columns, which includes extra columns, in each OR gate of a fixed OR plane, the total number of columns is '*m*' = '*8*' x '*y*' as shown in Figure 4. It is the same as the length of SCR2.

The MAP stores original personalities of data being programmed to the AND array. It is the same as the information of the Fuse Map. A MAP shown is for an AND plane of a block, thus this MAP is a '*n*' x '*m*' matrix. The SAP is exactly the same size array as the MAP. It has information about actual state of each cross-point of an AND plane after testing. It will show the state of stuck-at faults on the cross-points. It includes both the actual state as well as the stuck-at fault information. It will be compared with the MAP to find and locate faults. The SCR3 and SCR4 load the data column by column from the SAP and the MAP, respectively. The Comparator compares these two data with each other. The operation of comparator can be realized in few ways. For instance, the comparator can apply the 'minus' operation. It subtracts SCR4 from SCR3 bit by bit. If the result of subtraction is '0', then the cross-point, corresponding to that bit is correct which means that it has no fault. If the result is '-1', then it has stuck-at-0 fault, and if the result is '1', then it has stuck-at-1 fault in the AND plane. The EXOR gate can be used to know the existence and location of faults, but it cannot tell whether the fault occurred is stuck-at-0 (s-a-0) or stuck-at1 (s-a-1). Thus, the proposed design uses the 'minus' operation circuit instead of EXOR logic. The MAP and the NC register data will be updated by these compared results to reprogram the AND plane. The SCR5 has the complementary data from the SCR2 by the inverters since the SAP should have exact information of each E^2CMOS cell (cross-point) after testing. MAP should be compared with the SAP that includes the inverted data from the SCR2. It will be described in more detail in section 3.3. The structure of MAP and SAP is shown in Figure 5.

Inputs	c 1	c 2	----------	c 8y-1	c 8y
r1	1	0	----------	1	1
r2	0	1	----------	1	1
⋮	⋮	⋮	⋮	⋮	⋮
r (n - 1)	1	0	----------	1	1
r (n = 32)	0	1	----------	1	1

r : Row c : Column 1 : ON 0 : OFF 8y = m

Figure 5. Structure of MAP and SAP arrays

If a certain bit has '1', the corresponding cross-point (E^2CMOS cell) is 'ON', and if a certain bit has '0', the corresponding cross-point (E^2CMOS cell) is 'OFF'.

The NC Register informs about the status of each column in each OR group. As shown in Figure 6, each row represents an OR gate, and a column represents the input of an OR gate. There is '*y*' number of columns in the NC since each OR gate has '*y*' number of columns. There is 'k' number of ORs in this NC. In the GAL16V8, '*k*' will be 8 since it has 8 OLMCs. The intersection of a row and a column being '0' in the

NC means that the corresponding column of that OR is being used. If it is '1', then that column is useful for reprogramming a new logic function or replacing a faulty column. It can be considered as available extra column to repair faulty columns. The '-1' means that the column corresponding to that location of the AND plane cannot be used to repair a faulty column or to reprogram a new logic function. This Self-Repair method will be explained in Section 4.

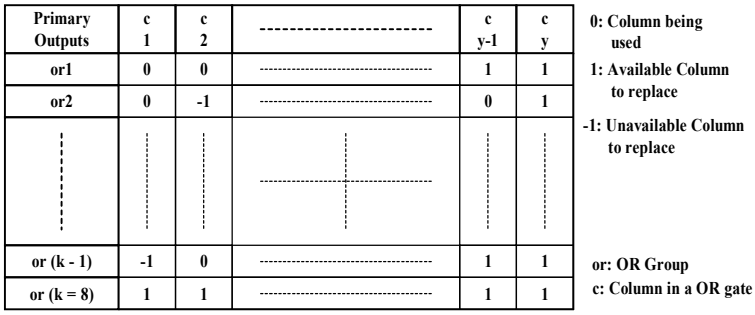


Figure 6. Structure of the NC register

3.3 Test generation and fault diagnosis/fault location

All stuck-at faults mentioned in Section 3.1 on cross-points of a GAL can be determined by a pattern. We have an universal test set for detecting faults. We use well-known walking 0 technology. This test vector set, which is provided by the FLFRP, can find whether the fault is a 's-a-0' or a 's-a-1'. It can determine the exact location in which the cross-point is faulty. All multiple faults of a column in an AND plane can be detected and located by this test set. The length of a test vector is n bits and the 'n' test vectors are needed to test if there are 'n' inputs because only one bit has '0' value in a test vector and it should affect each row. It detects the faults row by row in the AND plane. The FLFRP initializes SCR1 to '011…11', and this initial test vector will be shifted 'n-1' times from left to right. In this first test vector, the first bit is only '0' and others are '1's. The value of '0' in the first bit is fed into the first row (input), and it will affect the cross-point that is programmed (connected) as a value of ON. In other words, the result of this AND gate should be '0', but if the output of this AND gate is a '1' it will be a s-a-0 fault. On the other hand, the value of '0' fed into the disconnected cross-point does not affect the output of the AND gate. However, if the output of this AND gate is a value of 0, it will be a s-a-1 fault. Thus, all faults are detected by this universal test set, and it is shown as follows.

Test Vector Set (Test Pattern)

1 2 3 … n-1 n (inputs)
0 1 1 … 1 1
1 0 1 … 1 1
1 1 0 … 1 1
. . . … . .
1 1 1 … 0 1
1 1 1 … 1 0

A simple example in which an AND plane has 16 cross-points (4 inputs times 4 columns) is shown in Figure 7. The procedure of generating the test set in SCR1, and

next storing the scanning result from SCR2 into the SAP is illustrated as an animated sequence in this Figure.

In Figure 7, the symbol ‘X’ denotes a programmed value of ON in the respective intersection of a row and a column. This example assumes that there are no faults in the AND plane.

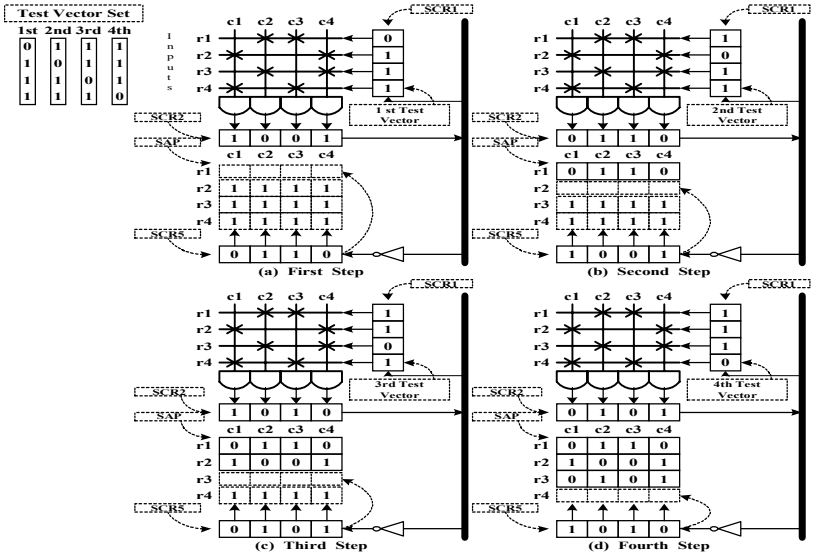


Figure 7. Generation of the test vector set/ storing scanning results into the SAP

The first test vector ‘0111’ generated by the FLFRP is stored in the SCR1. This vector is fed into each input line as shown in Figure 7. The first bit of the SCR1, ‘0’ cannot affect the output of the first AND gate since the cross-point of c1 (the first column) and r1 (the first row) is OFF. Thus, the first bit of SCR2 has ‘1’. The second bit of SCR2 has ‘0’ since the first bit ‘0’ of the SCR1 is fed on the second AND gate according to ON of cross-point of c2 and r1, and so on. The scanning result ‘1001’ of the SCR2 is transmitted through the diagnosis/repair bus to the SCR5 after inverting this data as shown in Figure 7. This inverted data ‘0110’ is stored on the first row of the SAP. This data ‘0110’ of the SAP is the same as the first one of the MAP since the first row has no faults. This ‘0110’ is also interpreted as the programmed state of that row. Figure 7 shows the procedure of generating the test set and storing the scanning result into the SAP.

Figure 8 illustrates the comparison operation for finding faults in a circuit without faults. Figure 9 shows the fault diagnosis and location of a simple example with faults.

After getting all data in the SAP, these data are compared with the MAP. For comparing operation in Figure 8, the SCR4 receives the c1 from the MAP, the SCR3 gets the c1 from SAP, the c1 of the MAP is subtracted from the c1 of the SAP, and so on. As shown in Figure 8, all results from this bit-by-bit ‘minus’ operation are ‘0’s, since there are no any faults in the AND plane. It also means that the MAP is exactly the same as the SAP according to all ‘0’s from the comparison operation. An example of a circuit with some faults is shown in Figure 9. The originally programmed personality of the AND plane is the same as in the previous example, but there exist multiple faults (s-a-0 faults and s-a-1 faults) in the AND plane in this case. All

multiple faults are diagnosed and located using the same method explained above. The bus is serial to decrease the pin-out of the chip.

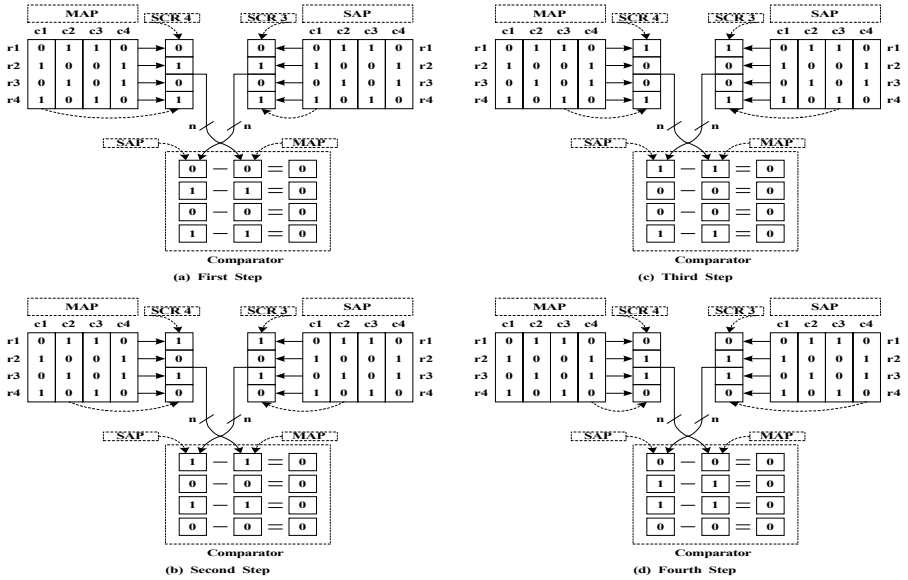
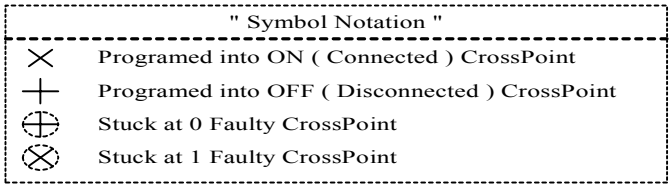


Figure 8. Comparator operation for finding faults on a circuit without faults

The symbolic notation that will be used in the following examples is the following:



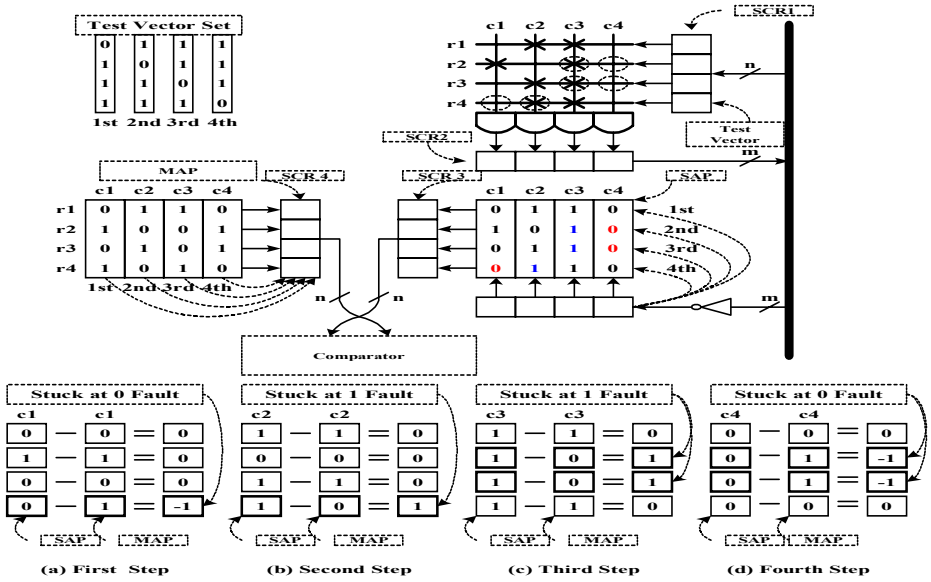


Figure 9. Fault diagnosis/location of an example with faults

In this case, there are multiple faults in the AND plane. The intersections of c3 & r2, and c4 & r2 have s-a-1 and s-a-0 faults, respectively. When a test vector '1011' is inserted from the SCR1, the result '0101' is obtained in the SCR2. The third bit of SCR2 will be '0' since that cross-point has s-a-1. It means that the second bit of the test vector, '0', can be transmitted to the third AND gate since that cross-point (E^2CMOS cell) is ON (s-a-1). The fourth bit of SCR2 will be '1' since that cross-point has s-a-0 which means that the second bit of the test vector, '0' cannot be transmitted to the fourth AND gate since that cross-point is OFF (s-a-0). Thus the diagnostic result '0101' of the SCR2 is obtained, and it will be inverted to store it into the SAP through the SCR5. The fault locating process is based on comparisons as shown in the above figure. The controller needs to obtain this result from the minus operation of the comparator to update and control the MAP and NC register. The detailed operation of the controller is not presented in this paper.

4. Self-repairing methods

A self-repairing method (column replacement method) is introduced in this section. This method should be applied after generating all test vectors and creating the SAP. Extra columns will be used to replace faulty extra ones even if the spare columns include faults.

The column replacement method starts after obtaining the SAP. First, it finds the column that has logic value '1' in the NC register. The faulty column is copied at the column location of the MAP corresponding to the column that has logic value '1' in the NC register. That column is reprogrammed into the AND plane according to the MAP, and the faulty column is reprogrammed with all '1s'. Therefore, the column that is reprogrammed with all '1s' cannot affect OR gate function that includes this faulty column. However, if all cross-points in certain column are faulty as s-a-0, it cannot be reprogrammed with all '1s'. Thus, it requires that at least one pair of literals of the same variables, for instance A and complement of A, must be reprogrammed as ON not to affect the OR function. It would be reasonable to have this assumption since it is extremely rare to have all s-a-0 in a certain column. Finally, the NC register is updated as '-1' for the reprogrammed faulty column with all '1s', and is updated as '0' for the replaced column. The column replacement method example with multiple faults is shown in Figure 10. There are 4 rows (input lines) and 8 columns (product terms) in an OR group of an AND plane. The originally programmed data (data of the MAP) is the same as in the previous example described in section 3.3. The symbolic notation is also the same as before. The numbered dotted box describes the sequence of this method. All cross-points that have not been programmed are initialized as logic values '1'. There are 4 extra columns (c5, c6, c7, and c8) initialized as '1' in this example. Thus, the MAP has logic value '1' on each location corresponding to those columns. The SAP was already obtained before applying the self-repairing method; thus it is not shown in this example. However, it can be described as just actual state of this AND plane, as shown in the figure. In Figure 10 (a), the MAP shows the initial state that is the information programmed for c1, c2, c3, c4 and the initial state of extra columns for c5, c6, c7, c8. The NC register has initial data which are '0' for c1, c2, c3, c4 since they are being used, and are '1' for c5, c6, c7, c8 since they are useful for replacing faulty columns.

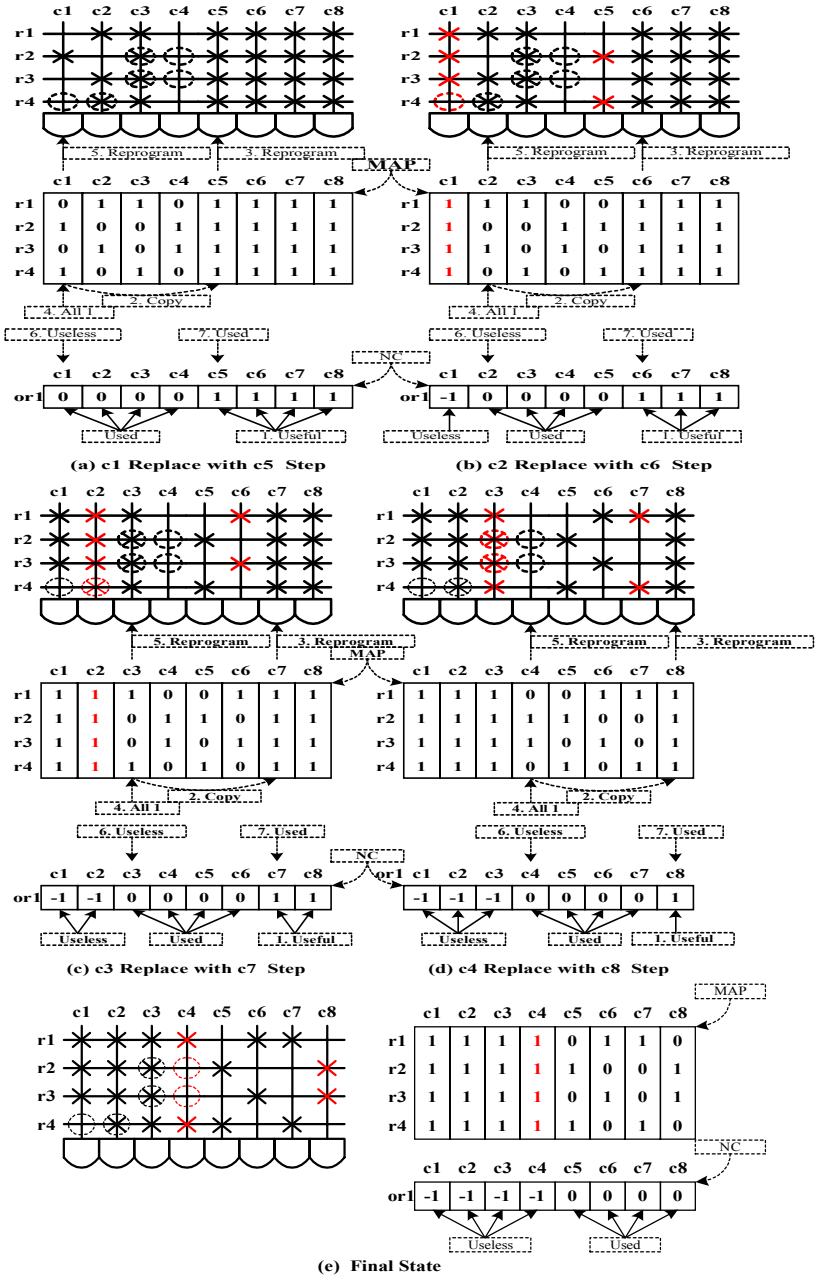


Figure 10. A column replacement method example of multiple faults

5. Evaluation methodology

In this section, simulation results are analyzed to evaluate the developed self-repairing method from the previous section and to prove that the self-repairable GAL will last longer in the field. The simulation program is programmed using Microsoft Visual C++ 6.0 with the GAL16V8.

The role of our evaluation methodology is to plan how to eliminate residual defects and confirm that outgoing products meet the product specification and are likely to be reliable.

Assumptions for an evaluation methodology are the following. Each cross-point in the AND plane has the same probability of stuck-at faults (s-a-0 and s-a-1) to occur. Consequently, each column and OR gate has also the same probability of stuck-at

faults to occur. The failure of one cross-point does not affect the likelihood that another cross-point will fail. Thus, the failure of one cross-point is assumed to be independent of the failure of another cross-point, likewise for each column and each OR gate. In GAL operation without self-repairing, if at least one cross-point stuck-at fault appears in a column, then that column will be faulty. This means that if at least one column fails in an OR gate, then that OR gate will be faulty, and if at least one OR gate is faulty in a GAL, then that GAL is useless. We do not consider the case when the number of extra columns is greater than the number of original columns because of large overhead.

Reliability is quality over time. Since time is involved in reliability, it is often measured by a rate. The most common measure of reliability for semiconductors is the failure rate, which is expressed in FITs (Failures In Time, Long Term Failure Rate) [14]. Failure rate estimates are based on the relationship between the failure rate of the device under working conditions. The FIT value gives the maximum number of failures that will occur in one billion (10^9) device-hours, with 60% confidence. A billion device-hours is roughly equivalent to ten thousand devices operating for ten years as well as the PPM (Parts Per Million, Early Failure rate) for one year [12,13].

The PPM, 0.05% (489/10,000 □0.05%) and FIT, 5.00% (5.07% □5.00%) of EEPROM obtained from National Semiconductor Corporation are adopted as the PPM and the FIT of a GAL in our simulation, referred to in section 1. In our simulation, the MAP is generated using 100% AND array utilization; thus all E^2CMOS cells are programmed using a random number generator. After creating the description of the MAP, the faults are simulated in an AND array by using a random number generator; the maximum number of cross-points that have faults at a time are limited to be less or equal to 5, 10, 20, 50, 100, 1000, and 2048. It is assumed that extra columns can also have faults. The extra columns will be increased in a multiple of 8 since there are 8 OLMCs in a GAL and each OLMC has the same number of extra columns. The results of the failure rate analysis are only valid under assumption that the assumed stuck-at fault model adequately represents all physical defects that can occur in a GAL device. Table 1 displays average looping time until a GAL is useless after running simulation in 100,000 times at each case. In order to calculate in Table 1 how many times the simulation looping time is increased in proportion to the number of extra columns, the ratios from Table 2 are used.

# of Extra Columns	# Of Faults Limit													
	≤5		≤10		≤20		≤50		≤100		≤1000		≤2048	
	Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL	
	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%
0	2708	25	2434	22	2299	21	2221	20	2198	20	2179	20	2179	20
8	7608	68	4800	44	3296	30	2553	23	2353	21	2191	20	2187	20
16	15094	136	8861	80	5314	48	3164	29	2613	24	2215	20	2199	20
24	24002	220	13772	125	7913	72	4089	37	2940	27	2248	20	2214	20
32	34400	315	19443	177	10917	99	5351	49	3412	31	2282	21	2225	20
40	46064	419	25754	285	14159	130	6720	62	4005	37	2320	21	2244	20
48	58607	537	32577	299	17791	163	8181	76	4793	44	2364	21	2265	21
56	72302	660	40030	367	21599	199	9793	90	5650	52	2402	22	2285	21
64	86989	792	47939	439	25732	236	11474	106	6541	60	2454	22	2303	21

Table 1. Average looping times of simulating the replacement methodology

These results show that having more extra columns, longer survival times in the field can be obtained (except two cases that the number of faults limit is less than 1000 and 2048 under 5% failure rate of a GAL (FIT)). However, these two cases will be very rare, even in less than 50 and less than 100 under either 0.05% or 5% failure

rate of a GAL. The results on 0.05% failure rate of a GAL are better than on 5% failure rate of a GAL.

If the average looping time can be converted to the lifetime of a GAL, it would provide how many extra columns a GAL should have in order to guarantee certain reliability and lifetime. Therefore, an ideal point could be estimated, where the maximum reliability of a GAL can be reached with the minimum cost. Finally, it proves that the self-repairable GAL will last longer in the field.

# of Extra Columns	# of Faults Limit													
	≤5		≤10		≤20		≤50		≤100		≤1000		≤2048	
	Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate Of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL	
	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05 %	5%
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
8	2.81	2.72	1.97	2.00	1.43	1.43	1.15	1.15	1.07	1.05	1.01	1.00	1.00	1.00
16	5.57	5.44	3.64	3.64	2.31	2.29	1.42	1.45	1.19	1.20	1.02	1.00	1.01	1.00
24	8.86	8.80	5.66	5.68	3.44	3.43	1.84	1.85	1.34	1.35	1.03	1.00	1.02	1.00
32	12.70	12.60	7.99	8.05	4.75	4.71	2.41	2.45	1.55	1.55	1.05	1.05	1.02	1.00
40	17.01	16.76	10.58	12.95	6.16	6.19	3.03	3.10	1.82	1.85	1.06	1.05	1.03	1.00
48	21.64	21.48	13.38	13.59	7.74	7.76	3.68	3.80	2.18	2.20	1.08	1.05	1.04	1.05
56	26.70	26.40	16.45	16.68	9.39	9.48	4.41	4.50	2.57	2.60	1.10	1.10	1.05	1.05
64	32.12	31.68	19.70	19.95	11.19	11.24	5.17	5.30	2.98	3.00	1.13	1.10	1.06	1.05

Table 2. The ratios based on the results from no extra columns

Now, we consider the hardware overhead in a GAL. In the measurement of area overhead, the LSI Logic Corporation's 0.5-micron LCA/LEA500K array-based products are used for synthesis [22]. All area measurements are expressed in cell units, excluding the interconnection wires. This measurement is separated into two parts, an AND plane and others (OLMCs, Input/Output Buffers/Inverters, a SCR1 (Serial-In-Parallel-Out Shift Register), and a SCR2 (Parallel-In-Serial-Out Shift Register)). In measurement of an AND plane size, each cross-point section consumes 1 cell, i.e. 2048 (32 inputs x 64 columns) cell units are measured for an AND plane in case of no extra columns. If there are 8 extra columns added in an AND plane, the AND plane has 2304 (32 inputs x 72 columns) cell units, and so on. The area overhead of other parts except an AND plane is measured by cell units given from [22]. The components used in the library are 2-input EXOR gate (3 cell units), D flip-flop (6 cell units), 1-of-2 multiplexer (4 cell units), 1-of-3 multiplexer (5 cell units), 1-of-4 multiplexer (6 cell units), tri-state buffer (3 cell units), inverter (1 cell unit), 2-input OR gate (2 cell units), and 4-input OR gate (3 cell units). For total area overhead of a GAL, size of an AND plane is added up OLMCs, Input/Output Buffers/Inverters, a SCR1 (Serial-In-Parallel-Out Shift Register), and a SCR2 (Parallel-In-Serial-Out Shift Register). For example, if there are 16 extra columns in an AND plane, the GAL has total number of 3618 cell units; 2560 cell units (32 x (64+16), AND plane) + 368 cell units (46 x 8, 8 OLMCs) + 18 cell units (Input/Output Buffers/Inverters) + 192 cell units (6 x 32, 32-bit SCR1) + 480 cell units (6 x 80, 80-bit SCR2)). Table 3 provides the area overheads and ratios.

These calculations have an acceptable error rate because GAL has a regular structure so that assumption of no connection overhead is realistic.

The ratios from Table 3 will be also used to calculate the performance defined as the ratio of looping time divided by the ratio of area overhead at each case. Table 4 shows the performance of a self-repairable GAL, depending upon the number of extra columns. This table represents the ratio of efficiency versus cost factor.

This result shows that all cases when the number of fault limits is less than equal 50, 100, 1000, and 2048 are not practical. The case of adding 64 extra columns for the assumption of the number of fault, less or equal 5 is the most efficient from Table 4 both under the 0.05% failure rate (14.15) and under the 5% failure rate of a GAL (13.96). Therefore, an ideal point could be estimated, where the maximum efficiency can be reached with the minimum cost from this table.

# of Extra Columns	# of Cells	Ratio
0	2410	1
8	3314	1.38
16	3618	1.5
24	3930	1.63
32	4234	1.76
40	4554	1.89
48	4858	2.02
56	5170	2.15
64	5474	2.27

Table 3. Area overhead and ratio

# of Extra Columns	# Of Faults Limit													
	≤5		≤10		≤20		≤50		≤100		≤1000		≤2048	
	Failure rate of a GAL		Failure rate Of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL		Failure rate of a GAL	
	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05%	5%	0.05 %	5%
0	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
8	2.04	1.64	1.43	1.45	1.04	1.04	0.83	0.83	0.78	0.76	0.73	0.72	0.72	0.72
16	3.71	3.63	2.43	2.43	1.54	1.53	0.95	0.97	0.79	0.80	0.68	0.67	0.67	0.67
24	5.44	5.40	3.47	3.48	2.11	2.10	1.13	1.13	0.82	0.83	0.63	0.61	0.63	0.61
32	7.22	7.16	4.54	4.57	2.70	2.68	1.37	1.39	0.88	0.88	0.60	0.60	0.58	0.57
40	9.00	8.87	5.60	6.85	3.26	3.28	1.60	1.64	0.96	0.99	0.56	0.56	0.54	0.53
48	10.71	10.63	6.62	6.73	3.83	3.84	1.82	1.88	1.08	1.09	0.53	0.52	0.51	0.52
56	12.42	12.28	7.65	7.76	4.37	4.41	2.05	2.09	1.20	1.21	0.51	0.51	0.49	0.49
64	14.15	13.96	8.68	8.79	4.93	4.95	2.28	2.33	1.31	1.32	0.50	0.48	0.47	0.46

Table 4. Performance of a self-repairable GAL

6. Conclusions and Future Works

The concept of a self-testing and self-repairing digital circuit has been presented using an example of GAL but it can be applied to FPGAs. The design, self-repairing, and evaluation methodology for GAL has been developed in this paper to self-repair faulty columns of a GAL. The faulty columns, assuming the stuck-at fault model, can be diagnosed and located by the proposed universal test vector. The fault-locating and fault-repairing architecture with electrically re-configurable GALs is the design method of automatic insertion of the testing, fault-locating, self-repairing circuit. This design method allows the repair of all multiple faults. A self-repairing methodology ("column replacement with extra columns") increases the performance of the device. This paper shows also an evaluation methodology, and it proves that the self-repairable GAL will last longer in the field. It presents also how many extra columns a GAL should have in order to guarantee the reliability and the lifetime of a GAL. Therefore, an ideal point and efficiency is estimated, where the maximum reliability can be reached with the minimum cost.

This paper is an initial attempt to present some of the possible approaches to design for self-repair. Although we present here only SOP (Sum of Product) type EPLDs, all ESOP (Exclusive OR Sum of Product) type circuits can be also used, possibly leading to even better results. This should be investigated in future research. The present approach could be also expandable for the PLAs or EXOR PLAs for

ESOP, GRM (General Reed-Muller), FPRM (Fixed Polarity Reed-Muller), and other AND/EXOR canonical forms and AND/EXOR multi-level circuits [23,24,25,26].

Our current work is to design a prototype FPGA that improves on the above approach, and compare the hardware overhead and the reliability for different methods of fault location and different architectures with the known methods of fault tolerant design.

References

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, "Digital Systems Testing and Testable Design," *IEEE Press*, pp.593-626, 1990.

[2] C. L. Wey and F. Lombardi, "On the Repair of Programmable Logic Arrays," *IEEE Trans. on Computers, Vol. 9*, pp.649-652, 1986.

[3] B. Avi, "A Tour of PLDs," http://www.ee.cooper.edu/course_pages/past_courses/EE151/PLD1, 1997.

[4] K. S. Son and D. K. Pradhan, "Design of Programmable Arrays for Testability," *1980 IEEE Test Conference*, pp.163-166, 1980.

[5] D. L. Ostapko and S. J. Hong, "Fault Analysis and Test Generation for Programmable Logic Array (PLA)," *IEEE Trans. on Computers, Vol. C-28, No. 9*, pp.617-627, September 1979.

[6] K. S. Ramanatha and N. N. Biswas, "An On-Line Algorithm for the Location of Cross Point Faults in Programmable Logic Arrays," *IEEE Trans. on Computers, Vol. C-32, No. 5*, pp.438-444, May 1983.

[7] J. E. Smith, "Detection of Faults in Programmable Logic Arrays," *IEEE Trans. on Computers, Vol. C-28, No. 11*, pp.845-853, November 1979.

[8] H. Fujiwara and K. Kinoshita, "A Design of Programmable Logic Array with Universal Tests," *IEEE Trans. on Computers, Vol. C-30, No. 11*, pp.823-829, November 1981.

[9] W. Daehn and J. Mucha, "A Hardware Approach to Self-Testing of Large Programmable Logic Arrays," *IEEE Trans. on Computers, Vol. C-30, No. 11*, pp.829-833, November 1981.

[10] R. Treuer, H. Fujiwara, and V. K. Agarwal, " Implementing a Built-In Self-Test PLA Design," *IEEE Design and Test*, pp.37-48, April 1985.

[11] S. C. Ma, "Testing BiCMOS and Dynamic CMOS Logic," *Center for Reliable Computing Technical Report, No. 95-1*, June 1995.

[12] National Semiconductor Corporation, "Quality Network – Failure Rates of Major Processes at National Semiconductor, National Semiconductor Failure Rate Trends, and National Semiconductor Reliability," http://207.82.57.10/quality/pages, May 1999.

[13] D. Sellers, "Quality and Reliability," *Quality and Reliability Hand Book – Space Electronics Inc.*, http://www.spaceelectronics.com/Spaceprod/reliability/qr.html, June 1999.

[14] "PLA and FPGA Devices," http://www.elec.uq.oz.au/~e3390/lectures/lect14/sld002.htm, 1998.

[15] "Sequential Logic Design with PLDs," http://www.elec.uq.oz.au/~e3390/lectures/lect14/sld014.htm, 1998.

[16] Lattice Semiconductor Corporation, "Lattice Semiconductor Data Book 1996," *Lattice Semiconductor Corporation*, pp.365-392, 1996.

[17] J. P. Hayes, "Fault Modeling," *IEEE Design & Test of Computers*, pp. 88-95, April 1985.

[18] W. Maly, "Realistic Fault Modeling for VLSI Testing," *Proc. of the 24th ACM/IEEE Design Automation Conference*, pp. 173-180, 1987.

[19] M. Marinescu, "Simple and Efficient Algorithms for Functional RAM Testing," *1982 International Test Conference*, pp. 236-239, November 1982.

[20] R. Nair, S. M. Thatte, and J. A. Abraham, "Efficient Algorithms for Testing Semiconductor Random-Access Memories*," IEEE Trans. on Computers, Vol. C-27, No. 6*, pp. 572-576, June 1978.

[21] F.G. Cockerill, "Quality Control for Production Testing," *1982 International Test Conference*, pp. 308-314, November 1982.

[22] LSI Logic Corporation, "LCA/LEA500K Array-Based Products Databook," *Document DB04-000002-03, Fourth Edition*, May 1997.

[23] R. Drechsler, H. Hengster, H. Schafer, J. Hartmann, and B. Becker, "Testability of 2-Level AND/EXOR Circuits," *Proc. of the 1997 European Design and Test Conference*, pp. 548-553, March 1997.

[24] A. Sarabi and M. A. Perkowski, "Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/XOR Canonical Network," *Design Automation Conference*, pp.30-35, 1992.

[25] A. Sarabi and M. A. Perkowski, "Design for Testability Properties of AND/EXOR Network," *Proc. of IFIP W.G. 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pp. 418-424, September 1993.

[26] T. Sasao, "Easily Testable Realizations for Generalized Reed-Muller Expansions," *IEEE Trans. on Computers, Vol. 46, No. 6*, pp. 709-716, June 1997.